**CarnegieMellon**
**Software Engineering Institute**

# Third Product Line Practice Workshop Report

Len Bass
Grady Campbell
Paul Clements
Linda Northrop
Dennis Smith

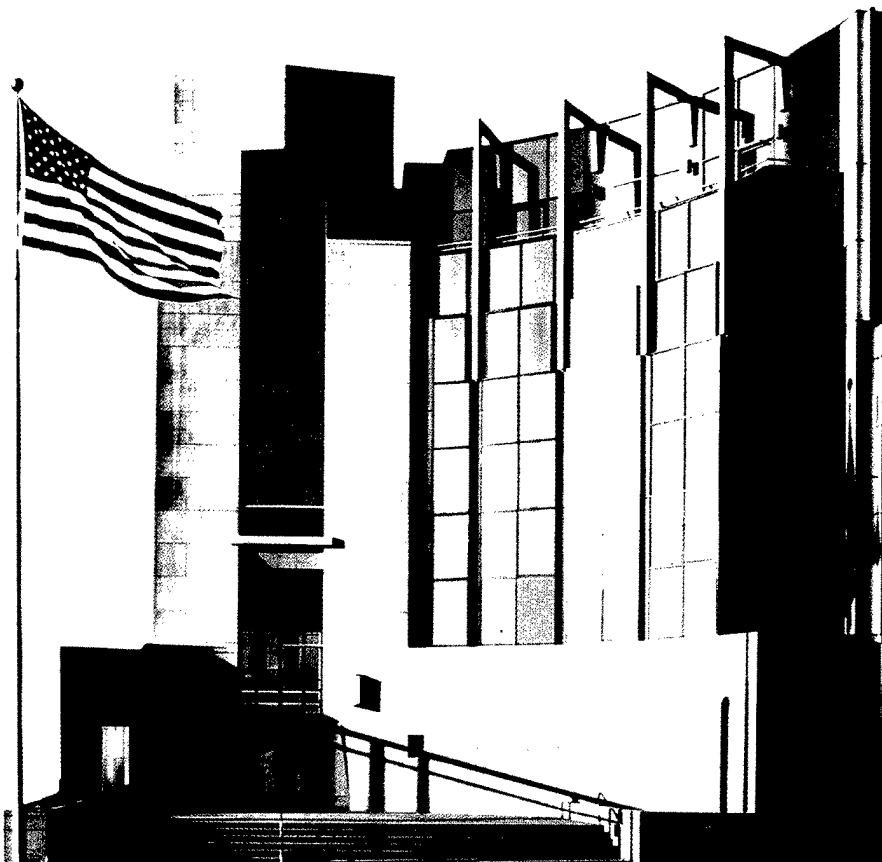*March 1999*

19990406 047

TECHNICAL REPORT
CMU/SEI-99-TR-003
ESC-TR-99-003

**Carnegie Mellon**
**Software Engineering Institute**

Pittsburgh, PA 15213-3890

# Third Product Line Practice Workshop Report

CMU/SEI-99-TR-003
ESC-TR-99-003

Len Bass
Grady Campbell
Paul Clements
Linda Northrop
Dennis Smith

*March 1999*

**Product Line Systems Program**
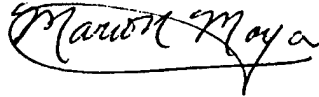
This report was prepared for the

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Mario Moya, Maj, USAF
SEI Joint Program Office

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work is sponsored by the U.S. Department of Defense.

This document is available through Asset Source for Software Engineering Technology (ASSET): 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone: (304) 284-9000 or toll-free in the U.S. 1-800-547-8306 / FAX: (304) 284-9001 World Wide Web: http://www.asset.com / e-mail: sei@asset.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218 / Phone: (703) 767-8274 or toll-free in the U.S.: 1-800 225-3842.

# Table of Contents

# List of Figures

# List of Tables

# Abstract

The Third Software Engineering Institute Product Line Practice Workshop was a hands-on meeting held in December 1998 to share industry practices in software product lines, to explore the technical and non-technical issues involved, and to evolve the SEI Product Line Practice Framework. This report synthesizes the workshop presentations and discussions, which described product line practices and analyzed issues in the areas of software engineering, technical management, and organizational management.

# 1 Introduction

## 1.1 Why Product Line Practice?

Historically, software engineers have designed software systems for functionality and performance. A single system mentality prevailed. Little attention was paid upfront to the consequences of a design that might be required in the production of multiple software-intensive products. Large software development, acquisition, and reengineering efforts undertaken with this single-system mentality perpetuate a pattern of large investment, long product cycles, system integration problems, and lack of predictable quality. Each product involves vast investments in requirements analysis, architecture and design, documentation, prototyping, process and method definition, tools, training, implementation, and testing.

An increasing number of organizations are realizing that they can no longer afford to develop multiple software products one product at a time: they are pressured to introduce new products and add functionality to existing ones at a rapid pace. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain market presence, compensate for an inability to hire, and leverage existing resources. Many organizations are finding that the practice of building sets of related systems together can yield remarkable quantitative improvements in productivity, time to market, product quality, and customer satisfaction. They are adopting a product line approach.

A *product line* is defined to be a group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission. It is most economical to build a software product line as a *product family*, where a product family is a group of systems built from a common set of assets.[1] In fact, the products in a software product line can best be leveraged when they share a common architecture that is used to structure components from which the products are built. This common software architecture[2] capitalizes on commonalities in the implementation of the line of products and provides the structural robustness that makes the derivation of software products from software assets economically viable. The architecture and components are central to the set of core assets used to construct and evolve the products in the product line. When we refer to a product line, we always mean a software product line built as a product family.

---

[1] A software asset is a description of a partial solution (such as a component or design document) or knowledge (such as requirements database or test procedures) that engineers use to build or modify software products [Withey 96].

[2] A software architecture of a computing system is the structure or structures of the system that consist of software components, the externally visible properties of those components, and the relationships among them [Bass 98a].

Some organizations refer to the core asset base that is reused on systems in a product line as a *platform*. Some organizations have other terms for product line. Terminology is not nearly as important to us as the underlying concepts involved, namely, the use of a common asset base in the production of a set of related products.

By product line practice, we mean the systematic use of software assets to assemble, instantiate, generate, or modify the multiple products that constitute a product line. Product line practice involves strategic, large-grained reuse as a business enabler. Some organizations have already experienced considerable savings in using a product line approach for software system production. Other organizations are attracted to the idea but are in varying stages of integrating product line practices.

In January 1997, the Software Engineering Institute (SEI) launched a technical initiative, the Product Line Practice Initiative, to help facilitate and accelerate the transition to sound software engineering practices using a product line approach. The goal of this initiative is to provide organizations with an integrated business and technical approach to systematic reuse so that they can more efficiently produce and maintain similar systems of predictable quality at lower cost.

One of the strategies to reach this goal involves direct interaction with and nurturing of the community interested in product line practice. This transition strategy has been executed in part by a series of product line workshops organized by the SEI. The workshop described in this report is the third such to bring together international groups of leading practitioners from industry to codify industry-wide best practices in product lines. The results of the previous two workshops are documented in SEI reports [Bass 97, Bass 98b]. The SEI has also refined the workshop results through work with collaboration partners, participation in other workshops, and continued research. In addition, the SEI is producing a framework for product line practice. The SEI's Framework for Product Line Practice describes the foundational product line concepts and identifies the essential activities and practices that an organization must master before it can expect to field a product line of software or software-intensive systems successfully. The framework organizes product line practices into practice areas that are categorized according to software engineering, technical management, and organizational management. These categories do not represent job titles, but rather disciplines. The framework is a living document that is evolving as experience with product line practice grows. Version 1 of the framework was made available on the Web in September 1998 [Clements 98].

## 1.2 About the Workshop

The SEI held the third in a series of two-day Product Line Practice Workshops in December 1998 to achieve the following goals:

- Share information and issues about product lines.

- Stimulate the growth of a network of interest in software product lines.

- Populate the framework with proven practices.

- Point out gaps where experience is not properly reflected in the framework.

The participants in this workshop were invited based upon our knowledge of their company's experience with strategic software reuse through product lines. The participants were sent a copy of the SEI's Framework for Product Line Practice to provide a common focus to structure the workshop presentations and discussions.

The workshop participants included
- Len Bass, SEI

- Grady Campbell, SEI

- Paul Clements, SEI

- Gary Chastek, SEI

- Sholom Cohen, SEI

- Kyle R. Fath, Cummins Engine Company

- Jean Jourdan, Thomson-CSF

- Philippe Lalanda, Thomson-CSF/LCR

- Ron Lannan, Cummins Engine Company

- Michelle Lohmeier, Raytheon Missile Systems Company

- Azza Mansour, Telesoft S.p.A.

- Linda Northrop, SEI

- David Sharp, Boeing

- Dennis Smith, SEI

- Nelson Weiderman, SEI

- Jim Withey, SEI

The domains in which the participants had product line experience include telecommunications, air traffic control, avionics, missiles, and engines.

A representative from each of the guest organizations was asked to make a presentation explaining that organization's approach to developing software product lines, describing how some of the practice areas in the framework are addressed.

On the second day, participant presentations were summarized. Then the participants divided into three working groups to explore the practices in software engineering, technical management, and operational management further. The working groups then presented their results to the entire group.

## 1.3 About This Report

This report summarizes the presentations and discussions at the workshop. As such, the report is written primarily for product line champions who are already working or initiating product lines practices in their own organizations. Technical software managers should also benefit from the information.

The report is organized into four main sections that parallel the workshop format:

- Introduction
- Product Line Experiences: Highlights of Participants' Presentations
- Product Line Practices and Issues: Working Group Reports
- Summary

The section following this introduction, "Product Line Experiences: Highlights of Participants' Presentations," synthesizes the product line experience of the workshop participants by describing common themes, pearls of product line wisdom, and productivity data. Section 3 is composed of the three working group reports on selected practices and issues in software engineering, technical management, and organizational management, respectively. The summary in Section 4 recaps the major themes and suggests future directions. Additionally, a glossary of terms is provided.

# 2 Product Line Experiences: Highlights of Participants' Presentations

The workshop began with presentations by the invited participants. As is the custom in these reports, this section will describe some of the highlights of those presentations so that readers can get a feeling for the tone and character of the workshop. As is also the custom, the intent is not to summarize the presentations exhaustively since the workshop's aim is to assimilate rather than merely record.

As noted in the introduction, one of the goals of each of these workshops is to validate the state of the SEI's Framework for Product Line Practice. All of the presenters felt that the SEI had identified the essential practice areas correctly. Although the working groups would result in some refinement, elaboration, and the identification of a small number of new practice areas, the participants by and large felt that the framework is well on its way to codifying the correct practices in a useful and high-fidelity way.

## 2.1 Common Themes

The importance of architecture was apparent in each presentation. Every speaker presented the architecture for his or her product line, underlining its central role. The creation of the right product line architecture was considered by all to be a crucial and difficult step to achieve. Several organizations are using the Unified Modeling Language (UML) as an architectural description vehicle. There were many references to deficiencies in UML that needed to be accommodated.

Another common theme of the presentations emerged for the first time at this workshop: one product line or many? No less than three of the organizations represented at this workshop had to make an explicit decision about whether their business area would be best served by a small group of tightly focused product lines, or a single somewhat more generic product line. For instance, Boeing produces avionics software for their F/A-18, F-15, and AV-8 families of fighter aircraft. Each of these aircraft comes in many versions: for example, various export configurations and various sensor and airframe configurations. Each aircraft could be viewed as a product line itself. Boeing therefore had to choose between three product lines or one. Raytheon had to choose between a product line for radar-guided missiles, a product line for infrared-guided missiles, etc., or a single product line for all missiles. Cummins similarly could have fielded a product line for their family of high-horsepower engines, another for their medium-horsepower engines, etc., or a single product line for all of their engines. Significantly, all three organizations decided in favor of a single product line, suggesting that the

benefits of product lines accrue faster when the scope is set somewhat broadly. Bottom line, scoping was a critical practice area for all.

Another common thread among our presenters at this workshop was that each of them has adopted the separate-group model for producing core assets. At the two previous workshops, we had identified two main organizational structures for product lines. In one, the core assets are built by a dedicated group, and products are built using these core assets by separate product groups. In the other structure, product developers produce core assets themselves on the way to building a particular product, with the caveat that those core assets must serve other product developers as well. At this workshop, all of the participants voiced support for the former model with the two separate groups.

A nice variation on this theme was carried out by one of our participants, who described a "lightweight organizational structure" that was put in place while the organization is transitioning to a product line. A research unit was tasked to help the business units coordinate and begin to work together to build and share core assets.

Other themes common to the presentations included

- the use of use cases as the primary mechanism for capturing requirements and domain models (Two participants included an explicit domain analysis in creating their product lines.)

- the judicious use of small prototypes and demonstration projects to work out the bugs in setting up a product line production capability in an organization

- a common timeframe for how long these organizations had been engaged in product line activities (Although some participants had had extensive experience in product line or product-line-like ideas, the projects they described were all two-five years old.)

The last point gave this workshop a somewhat different flavor than the two previous workshops. Because the product lines represented here were fairly young, setting up the product line was fresh in the minds of the presenters. Hence, their presentations tended to be more about their experiences in starting a product line for software-intensive systems, and less about experiences in running or reaping the benefits of one. (It will be seen in the working group discussions that launching a product line was actually thought by some of these participants to be a new and important practice area that the SEI framework had not yet accommodated.)

## 2.2 Pearls

Each presentation contained one or more "pearls" of knowledge that are worth repeating. Here are some of them:

- Develop a "lightweight organizational structure" to serve during product line adoption in order to help coordinate and set up practices, and "lightweight processes" to support this

organizational structure. These lightweight approaches facilitate smoother tranisitions to product lines that can be changed quickly and nimbly as the organization learns what does and does not work well for its own particular situation.

- Explicitly catalog the commonalities and variabilities among the products in the product line. For the architecture, commonalities include a common style, common components, common control, and data flow patterns. Mechanisms for variation include variation points, as well as having optional modes, threads, tasks, and services. While capturing commonalities and variations is at the conceptual heart of domain analysis and product line scoping, few organizations have reported doing so explicitly and to the degree of this participant.

- Comprehensively catalog the requirements of the products in the product line, so that commonalities and areas of incompleteness can be quickly identified. This essentially serves as an information model for the entire enterprise.

- Explicitly choose architectural styles and patterns (for example, layering or model-view-controller) to achieve desired quality attributes, and to describe and document the architecture using carefully partitioned structures or views (such as module, process, or uses views [Bass 98a]). In our view, archiectural views and structures are the carriers for the quality attributes that are the architectural imparts to the system, and these quality attributes are critically important in a product line. The architecture must enable sufficient performance, reliability, and most of all, modifiability across all members of the product line. Paying careful attention to the views and structures will help ensure that the quality requirements will be met.

- Don't establish the asset base as common across all applications (that is, requiring that every asset be used in every product). This results in a "least common denominator" kind of asset base, which is not nearly so useful as a set of core assets that can be largely used in a fair number of products.

- Recognize automatic regression testing as a critical capability when fielding a core asset base.

- Recognize the importance of requirements traceability in the core asset base, so that product builders can use the assets with confidence. One organization reported that they include requirements traces as part of their core asset base. Note the wording: They did not report that requirements traces were distributed along with the core assets, but that requirements traces *were core assets themselves*. This dovetails perfectly with our tenet that core assets include much more than just software components

- If needed, actually stop the efforts on stove-piped systems to initiate a product line. One organization stopped four stove-piped systems to launch a product line and achieved greatly improved time to market.

Finally, one participant reported on extensive work on building a model for a product line organization. This model takes into account both schemes mentioned above, and identifies and extensively describes the roles of product line staff.

Each of the three workshops seems to produce a new term that the participants warm to. At the previous workshop, it was the poetic "green-field effort" that describes designing a product line from scratch. At this workshop, two wonderful new terms were introduced. First, our engine software guests provided the colorful phrase to describe the tension between hardware

specialists (in their case, the engine builders) and software experts: "pistonheads versus digitheads." But perhaps of more lasting significance, two people referred to the circle-and-line drawings that inevitably are put forth to describe architectures as "architecture cartoons," a very apropos description that captures the essential fact that such drawings are at best incomplete renditions of the design.

## 2.3 Productivity Data

Each of the SEI workshops seems to uncover a breathtaking productivity statistic attributable to adoption of the product line strategy. At the first workshop, CelsiusTech reported using a single engineer to handle the integration testing for a 1.5 MSLOC (million source lines of code) Ada system, and porting such applications to a new operating system in just a couple of weeks. At the second workshop, Hewlett-Packard reported a six-fold productivity improvement achieved in three years, as measured by feature density per product shipped per engineer per unit time. This workshop was no exception. Cummins reported that the process of building a software system capable of running a new engine, which has taken on the order of a year, can now be done in about three days given their new product line approach.

# 3 Product Line Practices and Issues: Working Group Reports

## 3.1 Software Engineering

This working group discussed two of the software engineering practice areas from the framework: requirements elicitation, analysis, and management in some depth, and component development in less depth.

### 3.1.1 Requirements Elicitation, Analysis, and Management

The problem with requirements for a product line is that there are too many of them. Over 7,000 requirements existed for one product from one of the attendees. Multiply that figure by the number of products in a product line, and the complexity of the problem increases dramatically. A product line approach requires an organization to distinguish between those requirements in common across the product line and those requirements specific to a particular product. The approach requires a capability to trace requirements not only to delivered code modules but also to the software assets used in development across the product line.

To explore practices that help to manage this problem, the group decided to place the problem in a concrete context. We posed four specific questions:

1. How are requirements used?
2. What kind of requirements are we discussing?
3. What development context are we considering, new development or redevelopment?
4. What is an example of a product line requirement?

We identified three purposes for a requirements practice for product lines:

1. Distinguish requirements that will be satisfied by software assets from those that will be satisfied by specific product development. The purpose is to partition requirements into two camps: those that will be met by core assets without variation, and those that are specific for the particular product under construction.
2. Provide engineers with instructions as to their task. Domain engineers use requirements to develop assets that provide common application behavior and to accommodate refinement that will provide specialized behavior; application engineers use these assets to develop software to meet specific customer needs.

3. Provide product customers with traceability so that they can verify that their requirements have been met. In product lines, there is a need to be able to trace product requirements back through either (or both) the core assets and any specific product assets.

We agreed that requirements include both behavioral and quality-related requirements, at both runtime and development time. Since the bulk of requirements (>90%) are behavioral (excluding performance, modifiability, and other qualities), we decided to focus on behavioral requirements.

We discussed two development contexts: green-field and plowed-field. A "green-field" product line development (from scratch) will need a different process for requirements than a "plowed-field" product line development (based on legacy systems). In a plowed-field context, systems exist that encompass the (initial) scope of the product line, and these systems become the original repository of requirements. The requirements process for a green-field development is inherently top down. That is, the requirements are derived through reasoning about the usage of the system and not from the results of any previous elicitation. The requirements process for a plowed-field development can be either top down (in which case, they must be verified against the requirements for the existing systems) or bottom up. That is, the requirements for the product line are a (long) list, and this list is generated by taking the union of the requirements for all of the individual products and developing an abstraction that expresses specific behavior in terms of variation points

An example that we will use through this section is the requirement for the generation of noise. Flight simulators must generate noise to simulate engine noise. The generation of noise, then, is a common requirement. On the other hand, every type of aircraft (and hence every type of flight simulator) generates a different kind of noise. The noises are different in pitch, in intensity, and in duration. How, then, is it decided that the noise generation is a common requirement that should be implemented by a software asset? How does the application developer determine if the software asset supports the generation of the type of noise needed in a particular flight simulator? How does this application developer convince the customer that the noises being generated are being generated correctly?

All working group members agreed that some type of abstraction of the requirements was required to answer the above questions. An abstraction of the common requirements for the product line (such as the feature model generated by FODA[3]) that accommodates later refinement can aid domain engineers in identifying and designing assets and aid application engineers in determining the suitability of an asset for a system. The application engineer can customize based upon variation points and point to the variation of the asset to demonstrate traceability of customer requirements to code modules.

---

[3] Feature-oriented domain analysis (FODA) is an early domain analysis approach developed at the SEI [Kang 90].

In practice, abstracting common requirements entails generalizing requirements that are common to all systems and describing how the particulars of each general requirement can vary. Our discussion about noise generation is an example of a generalized requirement. The abstraction may contain generalized requirements that are specific to particular systems and does not necessarily contain concrete details about requirements that vary from system to system. Using the noise example, the abstraction might say that the generation of noise is a requirement for all systems; the generation of noise is related to engine conditions, atmospheric conditions, and the body type; and it might even say that the specific noises generated vary by pitch, intensity, and duration. The abstraction initially, at least, does not have to say which values of pitch, intensity, or duration are required for a particular system. These are variation points that are elaborated during product development.

We discussed whether the abstraction would include an enumeration of particular solutions as well as detailed variations in requirements. Using the noise example, this would entail describing the functions (and associated software assets) used to specify the noise for various conditions. Such enumerations are useful but are also expensive to develop. One strategy for evolving the abstraction is not to include the enumerations initially but to add them as application systems are developed. So the variation point would be there, but the exact variations would be evolved.

The problem with developing such an abstraction is that it has a cost. In a green-field development, the requirements must be dealt with in an abstract fashion in any case, so that the cost is unavoidable. Developing abstractions and understanding the relationships among them is the heart of the analysis and design processes. Capturing this understanding in a model of the requirements is not a large additional task. Thus, for doing a green-field development of a product line, an abstract representation of the requirements and their relationships is both useful and not terribly costly.[4] The situation is less clear for plowed-field development. In this case, the development of an abstract representation of the requirements is an additional task that incurs cost. Whether to develop an abstract representation of the requirements that accommodates variations then becomes a business decision where the cost of such an abstract representation must be weighed against the benefits of having one (or more likely, the cost of not having one). This decision will depend on a variety of local factors.

Once an abstract representation of the requirements is available, then the three purposes of requirements practices can be revisited in terms of this representation. Figure 1 illustrates.

---

[4] A more complete approach compliant with the feature model of FODA associates abstractions with specific refinements. This approach ensures more reuse opportunities of assets by capturing every feature in the product line. The experience of one of the participants revealed that the costs of this more complete approach are not dramatic and the benefits are dramatic. However, others have found the cost of this kind of complete elaboration prohibitive, and not feasible due to lack of upfront information about the potential variations.
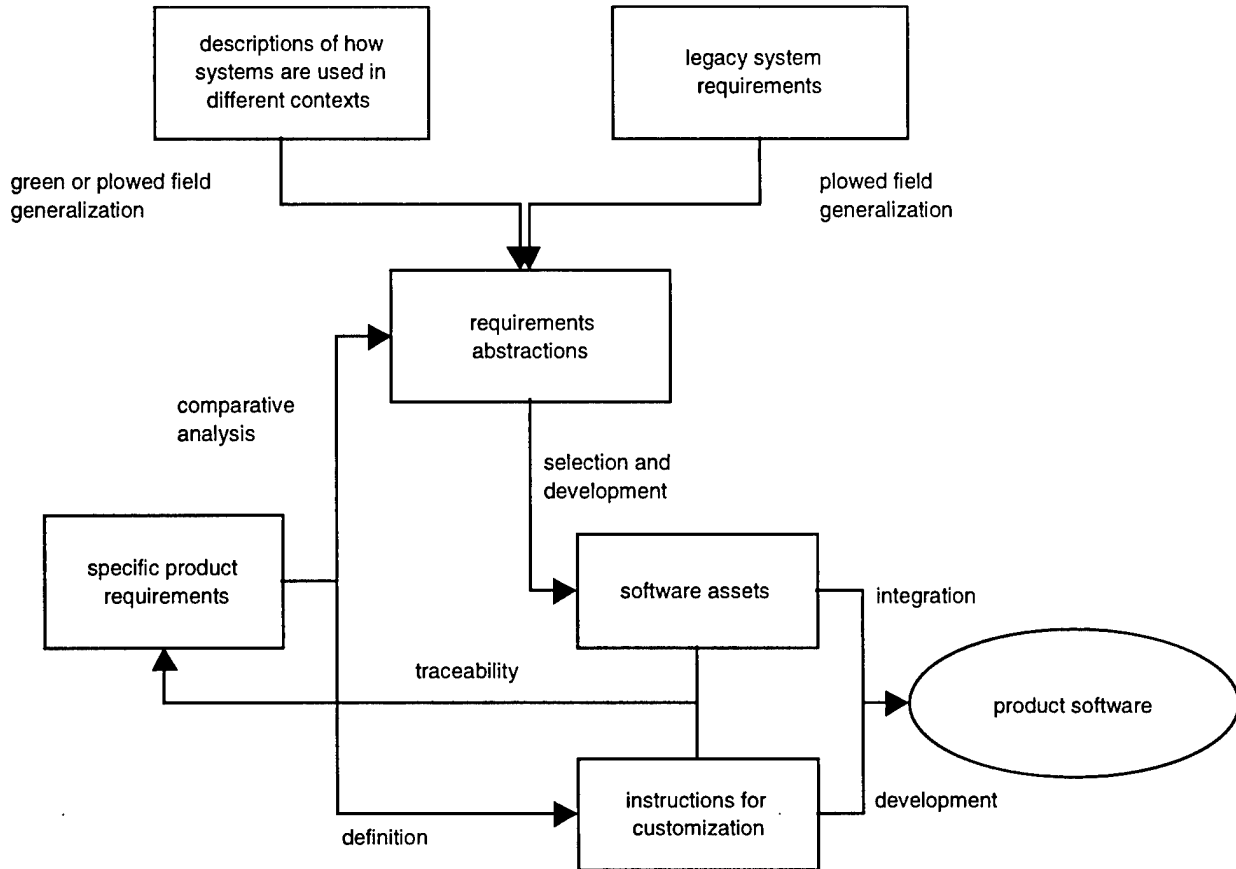
*Figure 1: Role of Requirements Abstractions in Product Line Development*

First, domain engineers would review the generalized requirements to select the common assets that should be developed. They would consider the ubiquity and complexity of the functions, the variation parameters that must be accommodated, and the skill required in asset customization. For each asset that is developed, the abstract and detailed requirements that are satisfied by the asset are allocated to that asset. Reviewing customer requirements, an application engineer would then be able to decide whether to use the asset, would know how to customize it, and would be able to trace final code to customer requirements. Using our noise example, again, there would be common components that had knowledge of the noise abstraction. These components might invoke other components that were written for a particular system and that added specificity to the noise requirements. To locate an asset, the application engineer would find the abstraction(s) that captures the specific noise requirement and select the components that were involved in satisfying the abstract requirement. Comparing specific product requirements to the components, the engineer could determine how to extend these components to satisfy the specific requirements. Asset extensions and other product requirements that are not satisfied by assets become instructions for product variations. If extensions to these components have been previously developed and are also catalogued somewhere (external to the abstract requirements, most likely), then the developer could examine this catalog and determine whether components existed that satisfied the current requirements.

Finally, to trace customer requirements, the application engineers would look at these components in a particular product and describe (at least at the component level) how a requirement is being met.

In summary, the complexity of product line requirements are managed by the development of an abstract representation. However, the development of the abstract representation has a cost that is best managed in a green-field development. The abstract representation can be used both for development specification and for traceability.

## 3.1.2 Component Development

The group discussed two aspects of the component development problem: (1) what is the process for developing components, and (2) what techniques are used to manage variation. We first discussed the process, and then we discussed the types of techniques used to manage variation.

The process discussion assumed green-field development of product lines but familiarity with existing systems in the domain. The first step in the process is to produce a proof of concept that important functions in the product line can be handled by common routines. This involves building components without concern as to their context. These components are viewed as prototypes and are intended to be thrown away, but these are still components that are being developed. The proof of concept fulfills two roles: it demonstrates feasibility, and it gives the developers some understanding of the architectural problems with which they must deal. Once some elements of the architecture are understood, then components that will live within that architecture can be developed.

Insofar as possible, components should be developed to be intelligent about their environment. That is, they should, either during initialization or as a portion of the build process, have the flexibility to adapt to different environmental conditions such as using different network protocols or different methods for being invoked or gaining input data.

Once the rules for writing components have been regularized, a guide describing the rules and the process for developing components should be written. This process guide is specific to the particular product line and includes knowledge of the architectural style.

The other topic briefly discussed was the technique used for managing variation or tailoring a core component for use in a particular product. One technique definitely not advised is to modify the components checked out from the core asset library. These components should be used "as is" without modification. If they must be modified, it suggests they were not written correctly. The reason is that any modification must be repeated every time a new release of the particular core asset component is incorporated into the product being developed. This results in additional work for every release of every modified component. If no modifications are allowed, then what techniques should be used? The group identified the following three techniques:

1. parameterization. The components have built into them knowledge of the variations possible and provide a parameter that controls use of these variations. This option assumes that the variability has been enumerated and, as noted in the section on requirements management, enumeration may be expensive and not possible.

2. inheritance and delegation. Object-oriented languages provide facilities for overriding default behaviors. The core asset component would implement a neutral behavior that then gets overridden for particular products. The group discussed, at length, the distinction between inheritance and delegation, and a strong opinion was expressed that delegation is much preferable to the use of inheritance. In any case, use of these features is a method for tailoring a core component to a particular product.

3. call backs. The core asset takes as a parameter the name of a function that should be called in certain circumstances. The function then gets written for each variation that is implemented. This is the technique used in many user interface toolkits. Call backs are actually an example of delegation, in this case "function delegation."

# 3.2 Technical Management

This working group focused on two of the technical management practice areas in the framework: process modeling and implementation, and risk management. The group discussed issues that are salient for the future definition of these practice areas. The results of these discussions are summarized in separate subsections below. In addition, the group provided overall comments on the other technical management practice areas, which are summarized in Section 3.2.3.

## 3.2.1 Process Modeling and Implementation

Because product lines represent a new way of doing business, those involved in developing or acquiring them require process guidance in the form of templates and examples. There is a need to define the following:

- the main practices and activities of product line development

- the flow among these practices and activities

- descriptions of how activities actually transform inputs into specific deliverables and products.

Figures 2.3 and 2.4 of the framework [Clements 98] (inserted here as Figures 2 and 3 below) illustrate, at a very high level, the processes for developing core assets and products, respectively. A process model is needed to provide the necessary detail to actually perform a set of tasks that constitute those high-level processes.

This process model should be created by process experts within the organization, rather than by experts in systems development. It is important to note that while guidance is needed to provide concreteness to an otherwise abstract set of tasks, this guidance must leave substantial freedom for implementation. For example, the process description should not prescribe an

organizational structure, since a specific structure may be inappropriate for an organization's culture.

The process description must be related to documents that describe how a product line is to be implemented in the organization, such as those outlining the product line's concept of operations. The concept of operations can be a vehicle to map the organizational structure to the process model.

### 3.2.1.1 Aspects Peculiar to Product Lines

Since the successful application of product lines is an emerging discipline, a number of issues related to effective process definition have yet to be resolved. Consequently, process modeling must be more flexible for a product line approach than for a single system product. When a process model is implemented, it should be done iteratively, and there must be checkpoints and a careful analysis of its effectiveness. Any generic product line process model will thus have significant variations between different organizations, depending on the process culture and product line maturity of the organization. In addition there will be variations within units of a single organization.

Product line process modeling must be managed flexibly, cutting across organizational units. The process definition should initially be more lightweight and have greater degrees of freedom. As an organization is successful with a product line approach, the process model can become more formal.

### 3.2.1.2 How Applied to Core Asset Development/Acquisition

Processes for core asset development will describe in detail the activities illustrated in Figure 2 (Figure 2.3, "Core Asset Development/Acquisition" of the framework). These processes need to be clearly defined and well tested since core assets are leveraged across a large number of potential products.

Initially, a generic process model can be developed that could then be customized for asset-based development for different product lines. Although product line development represents a distinct break from development of single systems, there are important approaches from single project development activities that can be adapted to developing core assets. For example, the method of use case modeling has been successfully adapted for domain analysis, which is an important practice area in developing the product space of a product line.
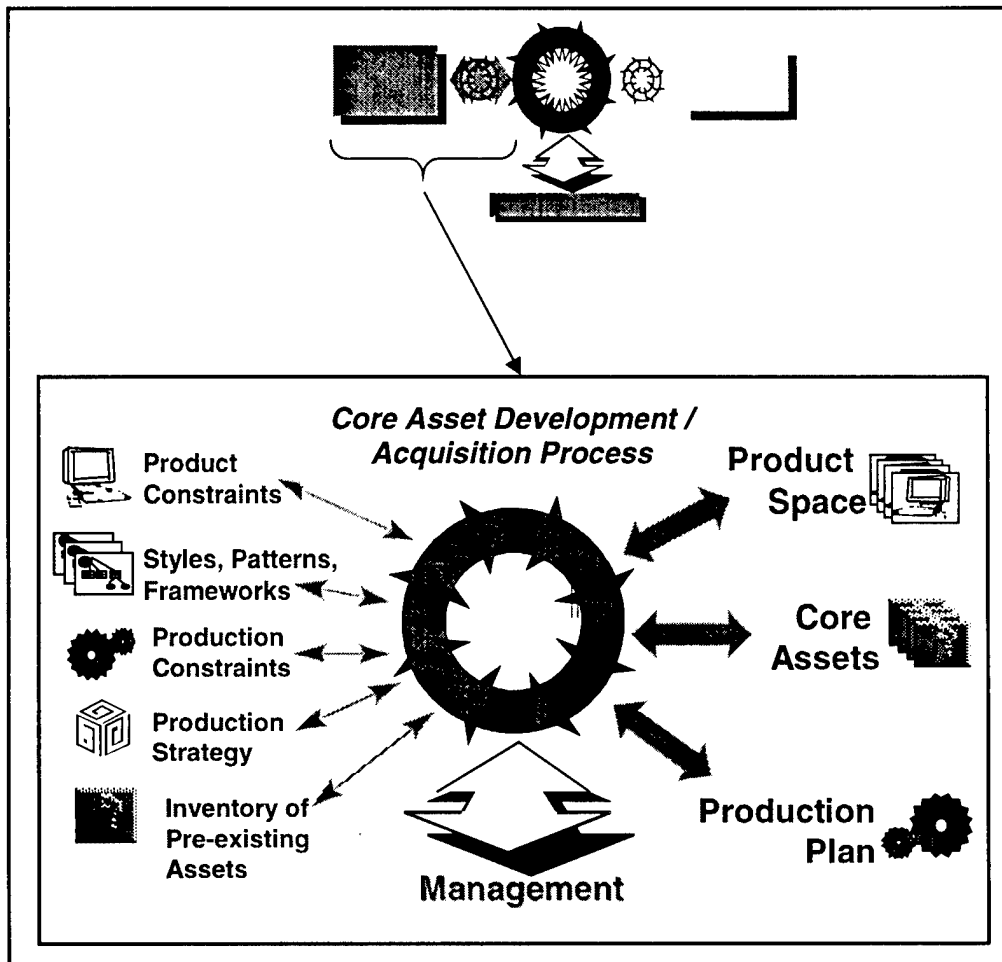
*Figure 2: Core Asset Development/Acquisition*

### 3.2.1.3 How Applied to Product/Acquisition Development

The process for product/acquisition development describes the activities of Figure 2.4, "Product Development/Acquisition" of the framework (Figure 3 of this report), and thus focuses on how to turn out a particular member of a product line from a set of core assets. The process needs to be more customizable than that of core asset development to accommodate variations both within components and between components. Variations will occur between different process models depending on the domain, the extent of core assets used for product development, the amount of automatic generation, or the amount of new code that is part of the product. Because this process involves specifying from a set of abstract requirements and assembling new products starting from a set of pre-existing core assets, a particular focus needs to be on multi-level testing and integration.

The process model for product development within a product line can be initially derived from a standard process model for single product development. It can be similar at a high level, with variations occurring at the detailed levels because of the use of existing assets.
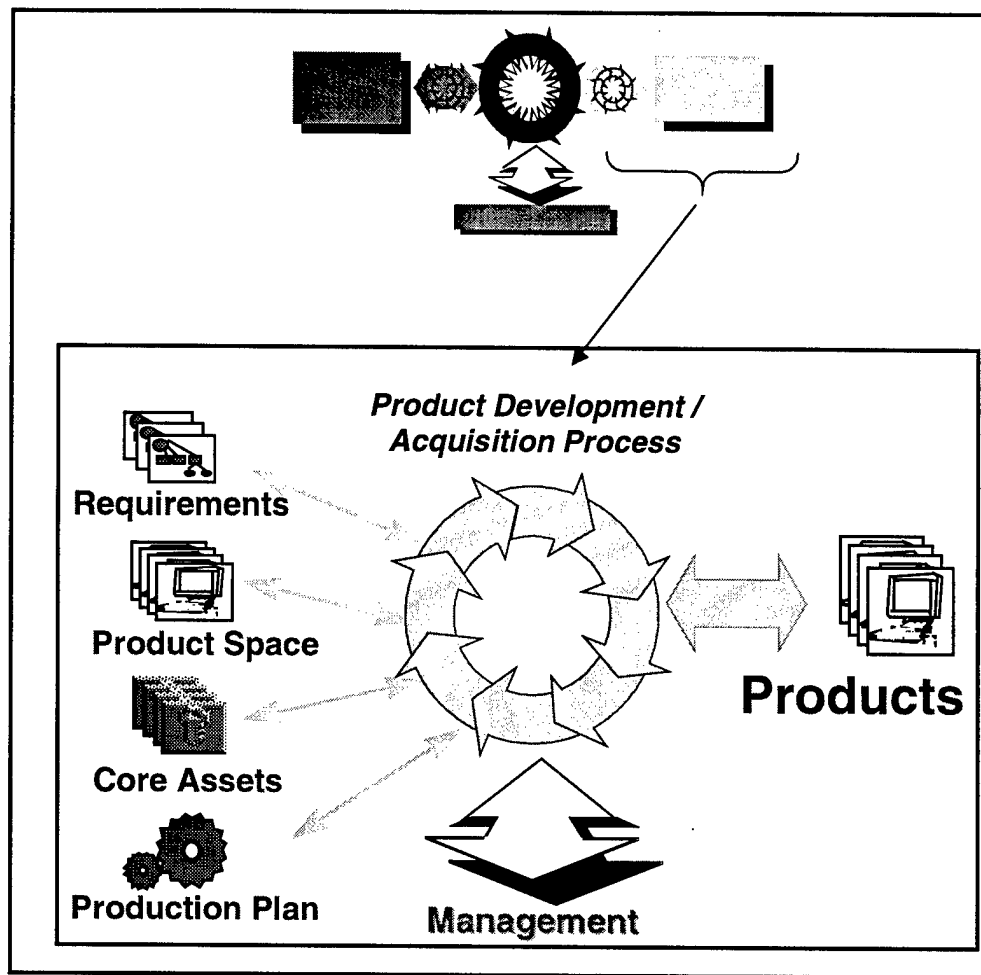
*Figure 3: Product Development/Acquisition*

## 3.2.1.4 Suggested Practice Guidelines

The working group suggested the following practice guidelines for developing a process model for product line development:

1.  Define the "to-be" (desired) state of the process model. Determine the delta of changes needed to get from the "as-is" state to the desired state. An "as-is" baseline can be obtained by surveying current practices in the organization. Input for the "to-be" state can be obtained from examples of successful applications of product line practices outside the organization [Brownsword 96, Bass 97, Bergey 98, Bass 98b].

2.  Develop an initial process definition and metrics to measure the effectiveness of the process. The process model should leverage standard case studies and handbooks [McFeeley 96, Willis 96]. If relevant, the process definition needs to conform with appropriate standards, such as the Capability Maturity Model (CMM®)[5] or the International Organization for Standardization (ISO). It also needs to be consistent with the process culture of the organization.

---

[5] Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

3.  Use a consistent and well-documented notation to define the process. The notation should have informal text to describe specific steps.

4.  Determine a manageable pilot on a small scale that will yield measurable results in a short period of time.

5.  Refine the pilot process based on results. Use standard approaches for the roll-out, such as insights derived from the IDEAL<sup>SM</sup> model [McFeeley 96].[6] The roll-out and adoption represent major tasks that need to be planned for appropriately.

6.  Roll-out the results on a wider scale.

7.  Provide training on the process.

### 3.2.1.5 Practice Risks

Risks include the following:

- There is a possibility of a process mismatch among a number of factors, such as the organizational structure, organizational culture, the process model that is employed, employees' experience and expertise, and the market to which the process model is applicable. For example, the defined process may be too heavyweight for the organization, resulting in a detailed set of practices that are not followed. On the other hand, the process model may be too lightweight and thus too general and at too high a level to provide practical guidance.

- The process may not accommodate a bidirectional flow between core asset development and product development processes.

- Because of divergent goals, skills, and backgrounds, there may be uneven quality in the contributions of the core asset teams and the product development teams. This may also result in a lack of harmonization of the process models for the different teams.

- The organization may not buy into the process model, resulting in a failure to adopt it.

## 3.2.2 Technical Risk Management

As with any complex software development or acquisition activity, risk management is crucial for implementing a product line approach. There are well-established approaches and guidelines for performing risk management that should be used as a starting point [Carr 93, Charette 90, Sisti 94]. These approaches provide practices for risk identification, analysis, planning, tracking, and control. For risk management targeted to product lines, the general practices will be consistent with the standard approaches. However, specific practices will require more coordination because of the range of organizational groups that are involved in product line activities and the fact that product lines often represent a different way of doing business.

---

[6] IDEAL is a service mark of Carnegie Mellon University.

### 3.2.2.1 Aspects Peculiar to Product Lines

The following aspects of risk management apply specifically to product lines:

- Product line risk management is complex because the risk mitigation plan will usually cross organizational boundaries and require buy-in and support from a number of diverse stakeholders.

- A product line approach can actually be viewed as a risk mitigator. The risks of low quality and poor integration between components are mitigated when new products are assembled around a proven set of core assets.

- Some types of risks will become more significant when a product line approach is used. For example, scoping represents a significant product line risk. If the scope is too large, the product line may become unmanageable. If the scope is too small, the leverage offered by a product line approach may not be obtained.

- In product line systems, there are inherent tradeoffs between generality as opposed to specificity. For example some core assets may be appropriate for a broad range of products, but the use of general core assets may hamper the satisfaction of specific product-level requirements in such areas as peformance and security.

- The risks identified by each product line practice area also need to be considered when performing a comprehensive risk assessment for the product line.

### 3.2.2.2 How Applied to Core Asset Development/Acquisition

The core assets have a ripple effect on each product in the product line. As a result, it is critical for core assets to have robust risk management to mitigate the risks of poor quality, schedule slippage, and defects that will have a cascading impact on specific products. In addition it is important that the core assets are scoped appropriately; the core assets should provide sufficient commonality to enable developing a product line, while permitting variation in product development.

### 3.2.2.3 How Applied to Product/Acquisition Development

Risk analysis and mitigation strategies performed by a product group must be communicated to the core assets group for an analysis of implications for core assets. In particular there is an inherent configuration management risk involving the relationship between core asset risks and product risks. Change requests reported at the core asset level require more analysis than change requests at the product level. However, there is a potential conflict between the time required for a full analysis at the core asset level and schedule needs at the product level. In addition, risk management for product analysis and acquisition must pay particular attention to real-time performance issues.

### 3.2.2.4 Description of Specific Practices that Apply to Practice Area

Specific types of good practices can serve to mitigate risks to product lines. These practices include

- use of architecture evaluations to identify tradeoffs and potential vulnerabilities in the architecture

- evaluation of the process and practices for product line development or acquisition (These evaluations may include formal assessments, such as CMM assessments, or more informal evaluations of specific practices. )

- reviews and walkthroughs of phases of the development and acquisition process

- configuration management audits, such as exercises to rebuild a system from old artifacts

- aspect-oriented programming to address tradeoff and design violations

### 3.2.2.5 Practice Risks

Built-in mismatches exist between the core asset and product cycles. These cycles are driven by sometimes conflicting goals, resulting in different priorities and different tradeoffs for decision making.

## 3.2.3 Discussion of Other Technical Management Practice Areas

The working group concluded with a brief discussion of the other technical management practice areas summarized in the following bullets:

- configuration management. The role of configuration management in product lines cannot be overstated.

- planning. Because product lines cut across products, planning needs to span multiple products and, in many cases, multiple organizational units. As a result, planning is complex and can include a number of critical paths.

- data collection and metrics tracking. Collecting data and tracking metrics must tie the metrics to business objectives and have operational measures of success.

- make/buy/mine analysis. A make-or-buy analysis must be performed throughout the life cycles of the development of both core assets and components for specific products. This analysis must focus on whether to develop new assets, mine existing assets, purchase commercial-off-the-shelf (COTS) products, or outsource the development of new assets. Some recommendations and issues for making these decisions are listed below:
  - If the required competency is strategic to the organization's business or mission, create the assets in house.
  - If the required competency is generic across an industry, purchase it.
  - The make/buy decision is strongly influenced by whether a development organization exists or whether the product line is essentially acquired from outside sources.

- If the development of the asset is outsourced, the issues of ownership and data rights must be addressed before finalizing the contract. At least two alternative providers ought to be available.
- A business case needs to be analyzed at the component level.
- The make/buy decision must be closely integrated with overall risk management and cost planning.

- program acquisition and management. The practice of program acquisition and management fits more appropriately as an organizational management practice area, rather than as a technical management practice area.

# 3.3 Organizational Management

The organizational management category of the framework encompasses product line practices that concern the adoption and institution of a product line approach by an organization as well as the overall operation of that approach. This working group attempted to clarify the nature of organizational management and the composition of its product line practice areas. Based on this discussion, the group decided to discuss three particular organizational management practice areas to a first level of detail for better understanding and any relevant insights: launching a product line, infrastructure planning, and proactive management. The group also discussed the form and content for describing risks in a practice area definition.

## 3.3.1 Context

The group felt a need to characterize the timelines that organizations follow in instituting a product line approach. The discussion focused on identifying the primary milestones of such a timeline.

### 3.3.1.1 Timelines for Product Line Institutionalization

There are two organizational management timelines to consider:

1. the *enterprise* timeline that charts how an organization institutes a product line approach

2. the *product line* timeline that charts how a business unit adopts a product line approach for a particular product line business

A timeline phase ends with a decision to initiate the next phase, to reinitiate the phase with different assumptions, or to terminate the associated effort.

An enterprise timeline has three phases:

1. evaluation

2. piloting

3. institutionalization

*Evaluation* involves an organization-level analysis of opportunities for a product line approach and a determination of the benefits and risks relative to current practices. *Piloting* se-

---

lects business units for early use and refinement of the organization's approach. *Institution-alization* establishes standards, training, and infrastructure capabilities for a standardized product line approach, including appropriate tailoring options.

A product line timeline has four phases:

1. assessment
2. launch
3. baselining
4. evolution

*Assessment* involves analyzing the viability and goals of a product line approach for a specific business unit scope based on assessments of market opportunities and technical capabilities. *Launch* attempts to develop business unit expertise in the product line approach and to create a preliminary product line strategy and capability to test the perceived viability and goals. A product line strategy, documented in a product line concept of operations, establishes an operational context consisting of product scoping, process model, business model, organizational structure, and environment. A product line capability consists of domain engineering resources and assets and a supported application engineering capability. *Baselining* involves creating a comprehensive product line strategy and supporting capability appropriate to the business unit's goals. *Evolution* entails making revisions to the baselined product line capability as business unit needs change over time.

A particular benefit of distinguishing these two timelines is the separation of the decision concerning an organization's approach to product lines from the decision concerning the approach's applicability to any particular business unit focus.

## 3.3.1.2 Nature of Organizational Management Practice Areas

The practice areas in the organizational management category should help an organization address

- the business (products and customers, now and in the future)
- the approach (product line practices that will benefit the business and how to transition to them)
- managing the approach (the management practices needed to ensure the success of product line efforts)

The business is adequately addressed by the folllowing current practice areas:

- Market Analysis
- Technology Forecasting
- Building and Communicating a Business Case

The approach is currently addressed by the following practice areas:

- Achieving the Right Organizational Structure
- Operations
- Developing and Implementing an Acquisition Strategy
- Customer and Supplier Interface Management

The group felt that three other practice areas relative to the approach should be considered:

- Process Planning
- Infrastructure Planning
- Launching a Product Line

Managing the approach is currently addressed by the following practice areas:

- Funding
- Risk Management
- Proactive Management
- Training

The group identified two other potential practice areas relative to managing the approach:

- Process Improvement
- Quality Management

The above discussion helped the working group establish a structure and clearer motivation for the organizational management practice areas and a basis for proposing new practice areas in this category.

## 3.3.2 Launching a Product Line

This practice area characterizes practices relevant to initiating a product line approach in a targeted business area. The group discussed concerns that need to be addressed when launching a product line and characterized each concern from the perspective of a single product line effort and from the perspective of the broader business enterprise of multiple product lines. The latter perspective has a motivation to standardize and improve how concerns are addressed for consistency and best practice across all the enterprise's product line efforts. Nine such relevant concerns are presented in Table 1.

| Concern | Product Line View | Enterprise View |
|---------|-------------------|-----------------|
| Training | Owning assets and process | Appropriate standardization and preparing new champions |
| Metrics and rewards | Exceeding business objectives | Rewarding beneficial product line efforts |
| Resource competition | Customer tradeoffs | Business case comparisons |
| Organization champion | Getting resources and support | Identifying candidates |
| Process improvement | Refinement of process and best practices | Standardization |
| Releases and upgrades | Stability and resources | Manageability of core capabilities evolution |
| Adoption planning | Supplying examples and lessons learned | Establishing expectations and asset sharing |
| Roles, responsibilities, and practices | Identifying appropriate people | Standardizing job descriptions across efforts |
| Funding | Obtaining funding | Establishing a business model and allocating funding |

*Table 1: Concerns When Launching a Product Line*

The group also discussed the following six techniques for limiting risks associated with launching a product line:

1. Limit scope (by focusing on a subset of work products).
2. Constrain tailoring.

3. Use metrics to evaluate the product line approach.

   a. asset utilization (as-is versus tailored)

   b. time to market

   c. quality (defects found, root cause, and location that indicates whether the problem is isolated or requires a fix to the process)

4. Coordinate management activities.

5. Align the culture and the organizational structure.

6. Focus effort on areas of early benefit and high-risk architectural elements.

## 3.3.3 Infrastructure Planning

The purpose of infrastructure planning is to determine what facilities are needed to support the product line approach and to plan their acquisition or development. The group identified the following seven concerns within the scope of this practice area:

1. information technology

2. legal support

3. library

4. connectivity

5. what to make available (including the capability needed to support the concept of operations)

6. tools

7. product definition support (including core asset releases)

## 3.3.4 Proactive Management

The purpose of the proactive management practice area is to specify the management activities needed for successful initiation of an effective product line approach. The group concluded that this practice area should have four goals:

1. Communicate product line strategy and vision.

   a. Communicate benefits.

   b. Provide championship.

   c. Fnd champions.

   d. Establish accountability.

2. Set objectives.

3. Create rewards and incentives.

4. Show commitment.

## 3.3.5 Risk Descriptions

As an adjunct, the working group considered the form of general risk information that is provided for practice areas in the framework. The group suggested that the framework might provide a more precise characterization of risks based on the concept of failure mode effects analysis [Charette 90].

Characterizing a risk in this form entails three factors:

1. probability of occurrence (How likely is failure of this sort?)
2. severity of impact (What expectations would change following such a failure?)
3. detectability (What monitoring, perhaps as measures, is necessary for a failure to be detected?)

A risk is relevant to an effort only if it may occur (neither impossible nor certain), its effects could increase costs or cause delay or failure of the effort, and there is some practical way to detect when such a failure has occurred.

The group identified six sources of risks as a guide for evaluating the completeness of risk management information for a practice area:

1. technical
2. management/leadership
3. human resources
4. tools
5. process
6. market

## 3.3.6 Conclusions

This working group started with the belief that organizational management is critical in identifying the business objectives and motivating the commitment needed for a successful individual product line effort. Organizational management oversight ensures that projects will work within a unified product line strategy for mutual success, rather than diverging into short-sighted efforts competing for scarce resources. Beyond this, organizational management is concerned with standardizing product line practices across the larger enterprise for optimum leverage of people and resources. Identified organizational management practices ensure success in repeatedly launching product lines across the scope of an organization's missions and markets.

# 4 Summary

The SEI's Third Product Line Practice Workshop explored the product line practices of technically sophisticated organizations with direct experience in software product lines. The presentations and discussions validated the pivotal pieces of the SEI's Product Line Practice Framework and provided additional content for currently identified practice areas as well as suggestions for new practice areas and other improvements.

This workshop featured a broad spectrum of application domains: embedded engine software, avionics, air traffic control, missiles, and telecommunications were all represented. This diversity increases our confidence in the assimilated results, because forging agreement across such diverse business areas suggests that we are indeed uncovering and codifying fundamental product line practices, as opposed to application- or organization-specific idiosyncrasies.

The motivation to embrace a product line approach was consistently voiced. That motivation included needs to

- achieve large-scale productivity gains
- improve time to market
- maintain market presence
- sustain unprecedented growth
- compensate for an inability to hire
- achieve systematic reuse goals
- improve product quality
- increase customer satisfaction

Other common themes also emerged: the importance of architecture and scoping, the separate-asset-group organizational structure, and the intricacies of launching a product line.

The working groups focused on specific practice areas within software engineering, technical management, and organizational management: requirements management, component development, process modeling and implementation, risk management, launching a product line, infrastructure planning, and proactive management. The empirical and anecdotal evidence that the workshop participants brought to the discussion significantly enhanced our current understanding of the practices and issues. New issues were uncovered, and insight was added

to many pervading issues that remain unsolved. Still a challenge for a product line approach is the repeatable integration of technical, business, and organizational practices.

The need for continued exploration and codification of both technical and non-technical product line practices remains. In an effort to grow both the information base and the community interested in software product lines, the SEI was encouraged by the participants to continue holding similar workshops and to continue reporting the workshop results to the software development community at large.

The results of this workshop are currently being incorporated into the framework,[7] which will continue to be refined and revised as the technology matures and as we continue to receive feedback and to work with the growing community of software engineers championing a product line approach. If you have any comments on this report and/or are using a product line approach in the development or acquisition of software-intensive systems and would like to participate in a future workshop, please send electronic mail to lmn@sei.cmu.edu.

---

[7] Version 2 of the SEI's Framework for Product Line Practice is planned for release on our Web site in August 1999. <URL: http://www.sei.cmu.edu/plp/framework.html>

# References

**[Bass 97]**         Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda; &
                      Withey, James. *Product Line Practice Workshop Report*
                      (CMU/SEI-97-TR-003, ADA 327610). Pittsburgh, PA: Software
                      Engineering Institute, Carnegie Mellon University, 1997. Available:
                      <http://www.sei.cmu.edu/publications/documents/97.reports/
                      97tr003/97tr003abstract.html>

**[Bass 98a]**        Bass, Len; Clements, Paul; & Kazman, Rick. *Software Architecture
                      in Practice*. Reading, MA: Addison-Wesley Longman, Inc., 1998.

**[Bass 98b]**        Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda;
                      Smith, Dennis; & Withey, James. *Second Product Line Practice
                      Workshop Report* (CMU/SEI-98-TR-015, ADA 345681). Pitts-
                      burgh, PA: Software Engineering Institute, Carnegie Mellon Uni-
                      versity, 1998. Available
                      <http://www.sei.cmu.edu/publications/documents/98.reports/
                      98tr015/98tr015abstract.html>

**[Bergey 98]**       Bergey, John; Clements, Paul; Cohen, Sholom; Donohoe, Patrick;
                      Jones, Larry; Krut, Bob; Northrop, Linda; Tilley, Scott; Smith,
                      Dennis; and Withey, James. *DoD Product Line Practice Workshop
                      Report* (CMU/SEI-98-TR-007, ADA 346252). Pittsburgh, PA:
                      Software Engineering Institute, Carnegie Mellon University, 1998.
                      Available
                      <http://www.sei.cmu.edu/publications/documents/98.reports/
                      98tr007/98tr007abstract.html>

**[Brownsword 96]**   Brownsword, Lisa & Clements, Paul. *A Case Study in Successful
                      Product Line Development* (CMU/SEI-96-TR-016, ADA 315802).
                      Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon
                      University, 1996. Available
                      <http://www.sei.cmu.edu/publications/documents/96.reports/
                      96.tr.016.html>

**[Carr 93]**      Carr, M.; Konda, S.; Monarch, I.; Ulrich, C.; and Walker, C . *Taxonomy-Based Risk Identification* (CMU/SEI-93-TR-006, ADA 266992). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1993. Available <http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.006.html>

**[Charette 90]**  Charette, R.N. *Application Strategies for Risk Analysis*: New York, NY: McGraw Hill, 1990.

**[Clements 98]**  Clements, Paul; Northrop, Linda M.; et al. *A Framework for Software Product Practice*. Available <http://www.sei.cmu.edu/plp/framework.html>

**[Kang 90]**      Kang, K.; Cohen, S.; Hess, J.; Novak, W.; and Peterson, A. *Feature-Oriented Domain Analysis (FODA) Feasibility Study* (CMU/SEI-90-TR-021, ADA 235785). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990. Available <http://www.sei.cmu.edu/publications/documents/90.reports/90.tr.021.html>

**[McFeeley 96]**  McFeeley, R. *IDEAL^{SM}: A User's Guide for Software Process Improvement* . (CMU/SEI-96-HB-001, ADA 305472). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available <http://www.sei.cmu.edu/publications/documents/96.reports/96.hb.001.html>

**[Sisti 94]**     Sisti, F and Joseph, S. *Software Risk Evaluation Method Version 1.0* (CMU/SEI-94-TR-19, ADA 290697). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1994. Available <http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.019.html>

**[Willis 96]**

Willis, R.R; Rova, R.M; Scott, M.D; Johnson, M.I; Ryskowski, J.F.; Moon, J.A.; Shumate, K.C.; and Winfield, T.O. *Hughes Aircraft's Widespread Deployment of a Continuously Improving Software Process*, (CMU/SEI-98-TR-006, ADA 358993). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available <http://www.sei.cmu.edu/publications/documents/98.reports/98tr006/98tr006abstract.html>

**[Withey 96]**

Withey, James. *Investment Analysis of Software Assets for Product Lines* (CMU/SEI-96-TR-010, ADA 315653). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available <http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.010.html>

# Glossary

**application engineering**
An engineering process that develops software products from partial solutions or knowledge embodied in software assets

**domain**
An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area

**domain analysis**
Process for capturing and representing information about applications in a domain, specifically common characteristics and reasons for variability

**platform**
Core software asset base that is reused across systems in the product line

**product family**
a group of systems built from a common set of assets

**product line**
a group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission area

**product line approach**
a system of software production that uses software assets to modify, assemble, instantiate, or generate a line of software products

**product line architecture**
Description of the structural properties for building a group of related systems (i.e. product line), typically the components and their interrelationships. The guidelines about the use of components must capture the means for handling variability discovered in the domain analysis or known to experts. (Also called a reference architecture)

**product line system**
a member of a product line

**production system**
a system of people, functions, and assets organized to produce, distribute, and improve a family of products. Two functions included in the system are domain engineering and application engineering.

**software architecture**  Structure or structures of the system, which consists of software components, the externally visible properties of those components, and the relationships among them [Bass 98a]

**system architectures**  software architecture plus execution and development environments

**software asset**  a description of a partial solution (such as a component or design document) or knowledge (such as a requirements database or test procedures) that engineers use to build or modify software products [Withey 96]

# REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

| 1. AGENCY USE ONLY (LEAVE BLANK) | 2. REPORT DATE March 1999 | 3. REPORT TYPE AND DATES COVERED Final |
|---|---|---|

**4. TITLE AND SUBTITLE**
Third Product Line Practice Workshop Report

**5. FUNDING NUMBERS**
C — F19628-95-C-0003

**6. AUTHOR(S)**
Len Bass, Grady Campbell, Paul Clements, Linda Northrop, Dennis Smith

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

**8. PERFORMING ORGANIZATION REPORT NUMBER**
CMU/SEI-99-TR-003

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
ESC-TR-99-003

**11. SUPPLEMENTARY NOTES**

**12.A DISTRIBUTION/AVAILABILITY STATEMENT**
Unclassified/Unlimited, DTIC, NTIS

**12.B DISTRIBUTION CODE**

**13. ABSTRACT (MAXIMUM 200 WORDS)**

The Third Software Engineering Institute Product Line Practice Workshop was a hands-on meeting held in December 1998 to share industry practices in software product lines, to explore the technical and non-technical issues involved, and to evolve the SEI Product Line Practice Framework. This report synthesizes the workshop presentations and discussions, which described product line practices and analyzed issues in the areas of software engineering, technical management, and organizational management.

**14. SUBJECT TERMS** domain analysis, product family, product line practice, reuse, software architecture, software platforms, software product lines

**15. NUMBER OF PAGES** 48

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |